Large Language Models for Wireless Networks: An Overview from the Prompt Engineering Perspective

Hao Zhou, Chengming Hu, Dun Yuan, Ye Yuan, Di Wu, Xi Chen,
Hina Tabassum, *Senior member, IEEE*, Xue Liu, *Fellow, IEEE*

Abstract—Recently, large language models (LLMs) have been successfully applied to many fields, showing outstanding comprehension and reasoning capabilities. Despite their great potential, LLMs usually require dedicated pre-training and fine-tuning for domain-specific applications such as wireless networks. These adaptations can be extremely demanding for computational resources and datasets, while most network devices have limited computation power, and there are a limited number of high-quality networking datasets. To this end, this work explores LLM-enabled wireless networks from the prompt engineering perspective, i.e., designing prompts to guide LLMs to generate desired output without updating LLM parameters. Compared with other LLM-driven methods, prompt engineering can better align with the demands of wireless network devices, e.g., higher deployment flexibility, rapid response time, and lower requirements on computation power. In particular, this work first introduces LLM fundamentals and compares different prompting techniques such as in-context learning, chain-of-thought, and self-refinement. Then we propose two novel prompting schemes for network applications: iterative prompting for network optimization, and self-refined prompting for network prediction. The case studies show that the proposed schemes can achieve comparable performance as conventional machine learning techniques, and our proposed prompting-based methods avoid the complexity of dedicated model training and fine-tuning, which is one of the key bottlenecks of existing machine learning techniques.

Index Terms—Large language models, wireless networks, prompt engineering

I. INTRODUCTION

As a sub-field of generative AI, large language models (LLMs) have received considerable interest from industry and academia [1]. The advancement of generative AI and LLMs also provides promising opportunities for 6G networks, including strong reasoning and planning capabilities, multi-modal understanding for 6G sensing, semantic communication [2], integrated satellite-aerial-terrestrial networks [3], etc. Despite the great potential, integrating LLMs into wireless networks still faces several challenges. Firstly, wireless networks are complex large-scale systems with various knowledge domains, i.e., signal processing and transmission, network architecture and design, protocol, standards, and so on. Applying general-domain LLMs directly to domain-specific network tasks may lead to poor performance. Secondly, the development of LLMs relies on high-quality datasets for fine-tuning adaptation, while there is a limited amount of high-quality networking datasets such as SPEC5G and Tspec-LLM [1]. Moreover, LLMs are extremely demanding in terms of computational resources. LLM pre-training and fine-tuning are usually implemented on high-performance GPUs such as NVIDIA A100 and H100, but wireless network devices usually have limited computational and storage capacities. LLMs involve a broad range of techniques, such as pre-training, fine-tuning LLMs for domain-specific tasks, retrieval augmented generation (RAG), prompt engineering, etc. Therefore, it is critical to identify an efficient method to better adapt LLMs to wireless networks.

Given the above opportunities and challenges, this work introduces prompt engineering, which is regarded as a resource-efficient and flexible approach to using LLMs with fast implementation speed [4]. These advantages will help to overcome the above LLM application challenges, e.g., deployment difficulties and requirements for computational resources. In particular, prompting refers to designing input prompts to guide pre-trained LLMs to generate desired outputs. It takes advantage of the inherent inference capabilities of pre-trained LLMs, and avoids the need for backward passes and gradient updates. Therefore, prompt engineering has several key features: 1) **Resource-efficient**: Prompting only needs forward passing of the model, and no need to store all intermediate activations for backpropagation. Such a resource-efficient approach may mitigate the computational burden of network servers and devices. 2) **Higher flexibility**: Prompting-based methods can quickly adapt to various tasks by crafting the corresponding textual demonstrations and queries without extra coding steps [5]. It indicates an efficient method to customize LLMs to address a wide range of network tasks. 3) **Fast implementation**: Prompt engineering relies on the inference capabilities of LLMs, and it avoids the time cost of updating LLM parameters. Therefore, the low response time can better handle low-latency network services. Finally, note that although prompting LLMs can significantly enhance the performance, it is not a standalone solution for LLM usage [6]. For instance, combining prompt engineering with fine-tuning may create a robust framework with both prompting flexibility and model specialization. However, this work focuses on prompting techniques to investigate their full potential.

LLMs have been discussed in several existing studies, but they mainly focus on system-level discussions and module designs, e.g., edge intelligence [7] and grounding and align-

Hao Zhou, Chengming Hu, Dun Yuan, Ye Yuan, Xi Chen, and Xue Liu are with the School of Computer Science, McGill University, Montreal, QC H3A 0E9, Canada. (emails:{hao.zhou4, chengming.hu, dun.yuan, ye.yuan3}@mail.mcgill.ca, xi.chen11@mcgill.ca, xueliu@cs.mcgill.ca); Di Wu is with the School of Electrical and Computer Engineering, McGill University, Montreal, QC H3A 0E9, Canada. (email: di.wu5@mcgill.ca); Hina Tabassum is with the Department of Electrical Engineering and Computer Science at York University, Toronto, ON M3J 1P3, Canada. (e-mail:hinat@yorku.ca).

TABLE I
SUMMARY OF VARIOUS LLM USAGE APPROACHES FOR WIRELESS NETWORKS.

Approaches	Key principles	Advantages	Potential difficulties	Computational resources & Time costs	Possible wireless network applications
Pre-training LLMs	1) Training LLMs from scratch on hundreds of billions of tokens for the next token prediction. 2) Pre-training is a foundational step of LLM development and usage.	Pre-training enables LLMs with comprehension, reasoning, and instruction following capabilities, which can address various downstream tasks.	Pre-training is extremely resource-intensive, requiring long training times, and substantial energy consumption and GPUs or TPUs.	High Thermal Design Power GPUs such NVIDIA H100-80G or A100-80G are required, e.g., training Llama3.1-405B took approximately 30.84M GPU hours.	A network-specific LLM indicates great benefits, but training network LLMs from scratch may be inappropriate due to the computational resource requirements.
Fine-tuning	Fine-tuning refers to adapting a pre-trained LLM to specific tasks by partially or completely updating the model weights or introducing additional adapter modules.	Fine-tuning significantly reduces time and computational cost compared to pre-training, producing LLM models tailored to specific tasks.	1) Fine-tuning has to consider the risk of overfitting and losing generalization. 2) Labelled data may be needed with additional human labour.	The cost may vary between model sizes and fine-tuning methods. If full fine-tuning a 7B parameter model with 8 A100-80G GPUs, it could be 100-200 GPU hours on a 100M token dataset.	Fine-tuning is a realistic approach to apply LLMs to wireless networks, adapting a general-domain LLM to specific tasks, e.g., network troubleshooting and configuration.
Retrieval augmented generation (RAG)	RAG combines LLMs with an external knowledge base, enabling the model to retrieve relevant information during inference and provide more accurate responses.	RAG can improve LLM's response accuracy by leveraging external knowledge sources, generating more up-to-date results.	1) RAG involves extra integration complexity, as it requires efficient retrieval algorithms. 2) RAG highly depends on a well-structured knowledge base, requiring dedicated creation.	1) The cost depends on the retrieval method and the size of the knowledge base. Inference may take seconds to minutes per query. 2) Requires a balance between retrieval speed and accuracy.	1) RAG is a more practical approach for implementing LLMs, especially when existing network-specific datasets are available. 2) It ensures the LLM has access to the most relevant and up-to-date network data.
Prompt engineering	Prompt engineering refers to crafting specific inputs to guide a pre-trained LLM to generate desired outputs, without any model retraining or tuning.	1) Prompt engineering has low computational costs and quick implementation. 2) It can be easily adapted to different tasks without additional training.	1) It requires expertise in crafting effective prompts. 2) It is also limited by the inherent capabilities and context window size of the pre-trained LLMs.	1) Without the need for backward passes and gradient updates, it requires minimal computational resources. 2) Time costs are significantly lower compared to other methods.	1) Prompt engineering enables rapid decision-making and responses in dynamic wireless network environments. 2) It aligns well with many network tasks.

[1] Note that the objective of this table is to introduce the features of each technique, and they serve different purposes in LLM development and application. These approaches can be combined to achieve better performance, e.g., fine-tuning network LLMs and then prompting.

ment [8]. Prompting engineering is also used for AI-generated everything services in [9], and our previous work also investigates LLM-enabled power control and traffic prediction in [10], [11]. However, this work is different from existing studies by systematically exploring prompt engineering and wireless network applications, providing detailed prompt designs and specific case studies. The main contributions are:

1) Firstly, we present in-depth analyses of LLM fundamentals and the feasibility of network applications such as pre-training, fine-tuning, RAG, and prompt engineering. Each technique is introduced in terms of computational resources and time costs, potential difficulties, and wireless network applications. Then, we provide an overview of a variety of prompting techniques and discuss their applications to wireless networks, such as in-context learning, chain-of-thought, prompt-based planning, and self-refinement.

2) We propose two novel prompting techniques, namely iterative prompting and self-refined prompting, aiming to address network optimization and prediction problems, respectively. Specifically, iterative prompting learns from previous experiences to improve LLM's performance on target tasks iteratively, which is suitable for handling network optimization problems. By contrast, self-refined prompting can correct their outputs through iterative feedback and refinement prompts, aiming to address network prediction tasks. These schemes rely on LLM's inference capabilities, and such resource-efficient techniques align well with the limited computational

capacities of many network devices. In addition, our proposed algorithms can be easily generalized to various wireless network applications, and the case studies show that they achieve satisfactory performance in network power control and traffic prediction problems.

II. LLM FUNDAMENTALS TOWARDS WIRELESS NETWORK APPLICATIONS

This section introduces various approaches for integrating LLMs to wireless networks, including pre-training, fine-tuning, RAG, and prompt engineering. Specifically, Table I presents these techniques in terms of their working principle, advantages, potential difficulties, computational resources and time costs, and possible wireless network applications.

Firstly, pre-training is a foundational step for LLM development, which includes data collection, filtering, and model training. Pre-training enables LLMs to develop fundamental capabilities such as comprehension, reasoning, and instruction following. Then, these capabilities are the foundation for developing other LLM-inspired techniques such as prompt engineering and planning, e.g., prompt engineering relies on the inherent capabilities of LLMs. However, it requires considerable computational resources such as NVIDIA H100-80G or A100-80G for implementation, and the training process may take weeks and months.

Given the training costs, pre-training a wireless-specific LLM from scratch can be extremely time and energy-
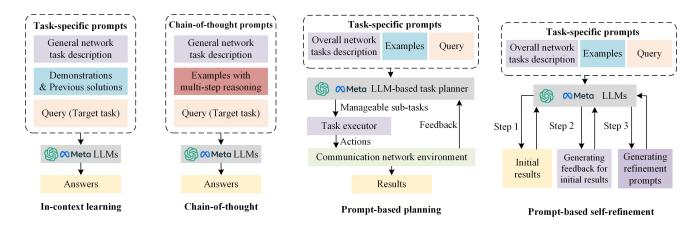
Fig. 1. Illustration of different prompting techniques.

consuming. By contrast, fine-tuning is a more affordable approach for adapting LLMs to network domains. Given pre-trained general-domain LLMs, fine-tuning indicates adapting LLMs to specific tasks by partially updating the model weights to fit smaller and domain-specific datasets such as 3GPP, 5G, and O-RAN standards. The fine-tuning costs are related to the LLM model size and fine-tuning methods. For instance, fully fine-tuning a 7B LLM model using 8 A100 GPUs may take 100-200 GPU hours for a 100M token dataset. For large-scale datasets in wireless fields, e.g., SPEC5G dataset with 134M words and 3GPP dataset Tspec-LLM with 534M words [10], the costs on computational resources may be larger. Similarly, RAG is another promising direction for practical LLM applications for wireless networks. It integrates LLMs with an external knowledge base, and then the LLM can retrieve related information from the external knowledge source during the inference. Since LLMs have access to the most relevant and up-to-date information, e.g., novel network standards, architecture and signal transmission techniques, RAG can significantly improve the generation quality.

Instead of updating model parameters, prompt engineering utilizes LLM's inference capabilities, designing specific input prompts to guide the generation of pre-trained LLMs. Importantly, prompt engineering avoids the need for backward passes and gradient updates of LLMs, and therefore the time costs are significantly lower than fine-tuning. Due to the low costs, it can quickly adapt to dynamic network environments, handling various network tasks with much lower response time. In addition, prompting engineering also aligns well with several crucial wireless network features. Firstly, many network devices are computational and energy resource-constrained, and prompt engineering has low requirements for computational resources, which may mitigate the energy consumption and computational burden on network devices. Secondly, prompt engineering allows human language-based instructions, e.g., network-specific task descriptions, questions, and solutions. By integrating natural language, network intelligence will become more accessible, especially for operators with minimal professional AI knowledge. In addition, prompt engineering

has high design flexibility. With proper textual templates, prompts can be efficiently designed using human language and applied to various network tasks, which also have minimum requirements on mathematical equations. Moreover, prompting can be easily combined with other techniques, e.g., fine-tuning LLMs on network-specific datasets, and then prompting the model to further improve the generated content.

On the other hand, prompt engineering also requires considerable expertise in crafting the prompts, which will directly affect the quality of generated content. Although prompt engineering can significantly improve LLM's performance, it relies on the model's inherent capabilities, e.g., achieved by pre-training or fine-tuning. Therefore, given the opportunities and challenges, it is crucial to explore the integration of prompting techniques and wireless networks, showing a promising way to utilize generative AI.

III. Prompt Engineering for Wireless Networks

This section presents several prompting techniques and their network applications, i.e., in-context learning, chain-of-thought, prompt-based planning, and self-refinement methods.

Firstly, as shown in Fig.1, in-context learning is a fundamental and crucial capability of LLMs. It refers to the process of learning from formatted natural language instructions and demonstrations, and then improving the performance on target tasks. Given contextual demonstration prompts, LLMs can address various downstream tasks by using existing knowledge from pre-trained data. Demonstrations are of great importance for in-context learning, serving as critical references for LLMs to learn from. Therefore, they should be carefully selected, formatted, and designed, which may require professional understanding and knowledge. In-context learning is a promising approach to applying LLMs to wireless networks, which can take advantage of previous network solutions as demonstrations to address unseen network issues.

Meanwhile, wireless networks are complicated systems, and one task may include several logic and reasoning steps. For instance, many network optimization problems include several elements, e.g., base stations, transmission power, bandwidth, and network users. Using simple input-output pairs

3

TABLE II

Prompting techniques	Key principles	Advantages	Potential issues	Wireless network applications
In-context learning	In-context learning empowers LLMs to learn from task-specific instruction and demonstrations. These prompts are designed with task-specific examples that are comprehensible to LLMs.	LLMs can be generalized to perform various learning tasks by utilizing existing knowledge from pre-training data and acquiring new task-solving strategies from contextual demonstration prompts.	Demonstrations greatly impact in-context learning, which should be carefully designed, e.g., selection, format, and order of examples.	In-context learning is a promising approach to applying LLMs to wireless networks, using previously accumulated solutions as input prompts to address new network issues.
Chain-of-thought	1) Chain-of-thought is an advanced strategy to improve LLM's capabilities in complex reasoning. 2) It can guide LLMs to better understand the logical steps from questions to answers.	LLMs can solve complex problems more transparently and logically through chain-of-thought demonstration prompts, e.g., "*Let's think step by step*".	Chain-of-thought prompting faces issues like incorrect reasoning and instability. Their inherent instability may provide incorrect and misleading intermediate reasoning steps.	Chain-of-thought can make LLMs better adapt to complicated wireless environments that require step-by-step reasoning and planning, e.g., project code generation with multi-step scheduling.
Prompt-based planning	Prompt-based planning aims to address more complicated scenarios such as multi-hop question answering. It breaks complex tasks down into manageable sub-tasks.	LLMs can make complex tasks more manageable by using their planning capabilities, lowering the overall difficulties.	LLM-based planning requires dedicated analyses to decompose a complicated task into multiple sub-tasks. Zero-shot and automated task decomposition is still the main difficulty.	Wireless networks are complicated systems and LLM-enabled automated task decomposition can be extremely useful for long-term network management and project development.
Self-refinement	Self-refinement allows them to correct their outputs through iterative feedback and refining demonstration prompts.	Through iterative feedback generation and prediction refinement, LLMs can self-enhance their outputs with strong effectiveness and scalable capabilities.	All historical outputs are iteratively appended as the inputs of LLMs, and it may lead to increasingly lengthy demonstration prompts and unavoidable resource consumption.	Self-refinement will significantly save human efforts for network tasks, since LLMs can constantly improve the previous network policy automatically.

as demonstrations cannot provide the logical reasons behind these examples, and LLMs may have difficulty in learning. To this end, chain-of-thought is proposed as an advanced strategy to enhance LLM's performance in complex reasoning tasks. Specifically, chain-of-thought will explain each logic step in the prompt demonstrations: "*Let's think step by step. Given the current base station transmission power* $\{BS\_power\}$, *using the Shannon capacity equation, the transmission data rate is* $\{user\_rate\}$. *Then, we compare the achieved rate with the target data rate, and* $\{user\_rate\}$ *is lower. Therefore, we may need to increase the transmission power.*" This example includes two steps, *"using Shannon capacity equation"* and *"compare the achieved rate with the target data rate"*, and then the final decision is to *"increase the transmission power"*. With these two-step explanations, LLMs can easily capture the relationship between transmission power and target data rate, and generate better replies using the logical chain.

In-context learning and chain-of-thought are mainly designed for single tasks, which may have difficulty tackling more complicated tasks, e.g., collecting information from multiple sources and producing multi-step reasoning to generate a comprehensive answer. Many complex network tasks need to be broken down into more manageable sub-tasks. For example, the network configuration usually involves multiple interrelated network elements, and network project development may consist of a series of coding and testing tasks. In this case, prompt-based planning aims to generate a group of sub-tasks, which will be executed one by one. This planning capability

is crucial for handling many large-scale network tasks.

In addition, LLMs may generate incorrect answers at the initial attempts, and external feedback is essential to improve its replies. As shown in Fig.1, self-refinement allows LLMs to self-improve their previous outputs by generating feedback and refinement prompts. For instance, the LLM's initial transmission power decision may be incorrect, and we can use the same LLM to evaluate and provide feedback, and then feed the feedback and refinement to the LLM, improving the initial power control decisions. In particular, it indicates that LLMs can automatically improve their previous outputs without human intervention. Self-refinement can significantly save human effort in improving LLM's initial replies, especially considering the complexity of network tasks.

Finally, Table II compares the above techniques in terms of main features, advantages, potential issues, and wireless network applications. Note that these methods can be combined to address complex tasks comprehensively. For instance, using self-refinement in prompt-based planning to save effort in the evaluation phase, and integrating chain-of-thought to improve the multi-step logic reasoning in sub-task decomposition.

IV. ITERATIVE PROMPTING FOR WIRELESS NETWORK OPTIMIZATION

Optimization techniques are of great importance for wireless network management, and this section proposes a novel iterative prompting scheme, in which we prompt LLMs iteratively to improve the performance on target network tasks.
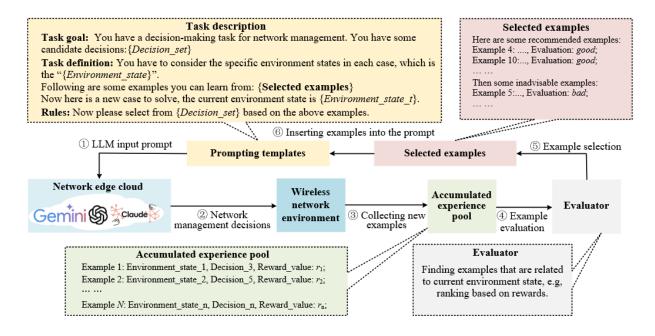
Fig. 2. Iterative prompting schemes for wireless network optimization

A. Natural Language-based Task Description

Firstly, as shown in Fig. 2, the target task is described by formatted natural language, including task goals, definitions, and extra rules, providing fundamental task information to the LLM. For instance, the task goal describes the problem as a "*decision-making task*", and it introduces the set of control variables as "*candidate decisions*". Then, it highlights the importance of "*environment states*", indicating that the LLM has to consider specific environment variables for decision-making. After that, it introduces the current target task by "*here is a new case to solve, and the current environment state is...*". Moreover, the final rules specify that "*select from {Decision_set} based on above examples*".

With these formatted task descriptions, the LLM can capture the key elements of the decision-making task, focusing on hidden patterns between environment states and corresponding decision variables. Such a design can also lower the understanding difficulties caused by professional network knowledge, since general domain LLMs such as GPTs and Llama are not specifically pre-trained for network applications.

B. Iterative Prompting Scheme

As illustrated in Fig. 2, the above task description will become input prompts for the LLM at network edge cloud, and the LLM selects a specific *Decision_n* from the set of candidate decisions {Decision_set}. Then, the network management decisions will be sent to the wireless network environment for implementation such as resource allocation, user association and transmission power level decisions. The network operation results be collected as a new example, including environment states, selected decisions, and rewards that are related to optimization objectives and network metrics.

After that, the examples from the network output, i.e., previous explorations, are collected in an experience pool, recording all previous examples. The examples in this pool will be further evaluated to choose specific examples well-suited for the target task. For instance, a straightforward strategy is to select previous examples with similar environment states as current states, e.g., similar user numbers and channel conditions. Then the previous decisions and rewards become useful references for LLM decision-making. However, note that many wireless environment states are continuous variables, e.g., channel state information or user-base station distances, and therefore it is unlikely to find examples with exactly the same states. Therefore, the evaluation and selection of examples are crucial in the proposed scheme.

Finally, the selected examples, i.e., recommended and inadvisable demonstrations, are integrated into the task description, providing useful references for LLM decision-making. Fig. 2 has presented steps 1 to 6 of the proposed iterative prompting scheme, and it can iteratively explore new decisions and accumulate new experiences. Therefore, the LLM can constantly learn from previous explorations, and then improve its decisions accordingly. Such an iterative prompting method can rapidly adapt to dynamic wireless environments.

Fig. 2 applies an exploration-exploitation strategy as in reinforcement learning (RL). However, iterative prompting has several advantages over RL approaches: 1) No model parameter updating and fine-tuning is required, which has much lower complexity than existing RL algorithms such as deep reinforcement learning (DRL). 2) It utilizes human natural language for network management. With minimum requirements on mathematical knowledge and equations, it makes network optimization more accessible than using RL. 3)

5

Demonstration prompt
* The hourly network traffic experienced 54.58,...,60.98 Mbps on *2013-11-05*.
* Based on the historical hourly traffic from 2013-11-05, what will the predicted hourly traffic be on *2013-11-06*?
* The predicted hourly traffic will be 50.34,...,67.12 Mbps on *2013-11-06*.

Query prompt   **Data prompt** The hourly network traffic experienced 56.48,...,71.70 Mbps on *2013-11-07*.

Based on the historical hourly traffic from 2013-11-07, what will the predicted hourly traffic be on *2013-11-08*?

first step: task description

LLM

second step: feedback generation   **third step: prediction refinement**

Data prompt The ground truth hourly network traffic experienced 52.32,...,65.45 Mbps on *2013-11-08*.

Predicted traffic The predicted hourly traffic will be 58.00,...,72.00 Mbps on *2013-11-08*.

Feedback prompt
* *Overall performance*: The Mean Absolute Error of the prediction is 8.92. The overall error should be decreased.
* *Periodical performance*: The function on the ground truth traffic is $f_{\text{act}} = -11.96\sin(0.89t - 8.11) - 23.55\cos(5.90t - 8.11) + 5.90$. The function on the predicted traffic is $f_{\text{pred}} = 274.02\sin(0.04t + 4.55) + 320.81$. The prediction should match the function of the ground truth traffic.
* *Format and completeness*: The predicted hourly traffic should be complete and match the format of the ground truth traffic.
* *Prediction method*: The prediction assumes minor adjustments based on the trend of increased traffic during peaking hours.

OK, let's predict hourly traffic on *2013-11-08* again using the above feedback. More accurate prediction method should be considered to improve the performance. The overall error should be decreased. The prediction should match the function of the ground truth traffic as closely as possible. The prediction should be complete each hour and match the format of the ground truth traffic.

Prediction improvement   **Refinement prompt**
* Early morning: a reasonable downward adjustment.
* Morning to early afternoon: increase during the peak morning and noon, based on the historical traffic.
* Late afternoon to evening: increase due to consistent periods until the evening.
* Night: a significant jump in late-night traffic, which might be event-driven or weekly behaviour.   **New predicted traffic**

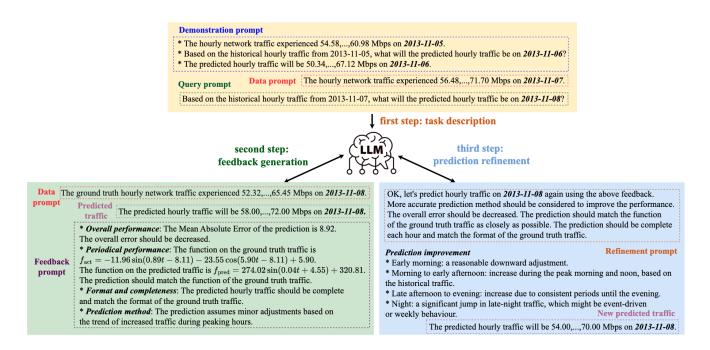The predicted hourly traffic will be 54.00,...,70.00 Mbps on *2013-11-08*.

Fig. 3. Self-refined prompting for network traffic prediction. As an example, by following prediction tasks framed in a question-answer format, the LLM is guided to predict hourly traffic prediction for 2013-11-08, using hourly historical traffic from 2013-11-07. Due to space constraints, we show traffic for selected hours only, while hourly traffic is fully utilized during interactions with the LLM. The specific feedback related to initial predictions is incorporated into the feedback prompt. The LLM further refines its predictions by following the actionable steps outlined in the refinement prompt. The process of feedback generation and prediction refinement is iteratively conducted until the prediction performance converges, whereas the inference is completed without engaging in the feedback generation and prediction refinement.

LLMs can provide reasonable explanations for their decisions, and help humans understand complicated network systems, which is beyond the capabilities of RL techniques. Finally, note that our proposed schemes can also be generalized to many other fundamental network optimization tasks, providing a novel LLM-based solution for network control.

V. Self-refined Prompting for Network Prediction

Prediction tasks are critical in wireless networks, using historical network data to predict future trends and behaviours such as network traffic, user demand, channel states, and device status. To showcase the potential of prompting techniques, this section presents a self-refined prompting method for network prediction problems, i.e., network traffic prediction.

A. Prediction Task Description

A crucial step in using LLMs for prediction problems is to convert numerical values into natural language sentences. Fig. 3 considers network traffic prediction as an example, and illustrates the demonstration prompt, data prompt, and query prompt. In particular, the traffic prediction task is framed in a question-answer format, adhering to a template-based description. The demonstration prompt is first incorporated to guide the LLM in making traffic predictions by leveraging contextual task descriptions for next-day traffic prediction. Then, the LLM predicts future traffic by responding to questions in the query prompt, based on historical traffic data provided in the data prompt, i.e., previous hourly network traffic.

B. Feedback Generation and Prediction Refinement

Initially, LLMs may generate incorrect predictions, and this subsection utilizes self-refined prompting to incorporate feedback demonstrations to provide comprehensive and valuable insights, aiming to improve the initial results [11]. As shown in Fig. 3, the feedback prompt is designed to encompass prediction performance, prediction format and completeness, and prediction method, which are shown as follows:

- **Overall performance**: The same LLM evaluates the overall prediction performance, e.g., Mean Absolution Error (MAE) between ground truth and predictions.
- **Periodical performance**: Considering that time-series data can be represented through sine and cosine functions in the real domain [12], the same LLM is tasked with projecting both ground truth and predicted traffic onto these functions, respectively. By matching the projection of ground truth, predictions are expected to capture the periodic nature of ground truth accurately, such as fluctuations during peak and off-peak traffic periods.
- **Format and completeness**: The predictions should match the format of ground truth as well as be complete for each timestamp, e.g., 24 values for daily traffic prediction.
- **Prediction method**: The prediction method is summarized in the feedback demonstration prompt, which calls for adopting more advanced and accurate methods to enhance performance further. To avoid additional computational costs, while the LLM may suggest developing

a neural network as a potential improvement method, this recommendation is disregarded.

Given the above instructions, the refinement demonstration prompts aim to provide specific actionable steps associated with feedback demonstration prompts. This allows the same LLM to self-refine previous predictions by adhering to the detailed feedback outlined in feedback demonstration prompts. Note that the process of feedback generation and prediction refinement is iteratively conducted until the prediction performance converges, whereas the inference is completed without engaging in the feedback generation and prediction refinement. Additionally, to prevent the repetition of previous incorrect predictions, the history of predicted traffic, feedback, and refinement demonstration prompts are also included as inputs to the LLM at each iteration.

In summary, LLMs excel at recognizing and performing new tasks through prompt engineering techniques that utilize contextual information, including task descriptions and demonstration examples. Specifically, the proposed self-refined prompting method designs task-specific demonstration, feedback, and refinement prompts, allowing the LLM to iteratively enhance its predictions by adhering to the instructions in these prompts, without requiring additional model fine-tuning or training. Compared to the dedicated training of long short-term memory (LSTM) networks, the proposed self-refined prompting method is more computationally efficient with the inference abilities, particularly advantageous for on-device learning on resource-constrained devices in real-world wireless networks. Given that iteratively appending all historical predictions can increase the length of demonstration prompts and potentially lead to higher resource consumption, a future direction for the self-refined prompting method will focus on effectively compressing extensive demonstration prompts, thereby further enhancing resource efficiency.

VI. CASE STUDIES

A. Simulation Settings

This section presents two case studies on network optimization and prediction problems.

1) **Network Optimization**: This case study involves a base station power control problem a fundamental optimization task in wireless network fields. The considered problem formulations aim to minimize the base station power consumption and meanwhile maintain an average data rate threshold for users[1]. We consider three adjacent base stations and the associated user numbers dynamically change from 5 to 15 each, the average data rate threshold is 1.5 Mbps/per user, and the channel gain applies 3GPP urban network models. This case study includes: ① **Iterative prompting**: We consider 2 LLM models: Llama3-7b-instruct as a small-scale model, and Llama3-70b-instruct as a large-scale model. We deploy the proposed iterative prompting technique as introduced in Fig. 2 and Section IV, in which the LLM will explore the environment, accumulate experience, and learn iteratively. ②
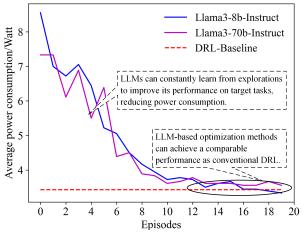
[1]Detailed equations can be found in [13] (equation (1.12) in Section 1.3.5)

DRL baseline: DRL is included as a baseline, which has been widely used to address network optimization problems after dedicated model training. We apply a classic deep Q-learning algorithm, in which the neural networks have 3 layers, the experience pool size is 10000, the batch size is 64, and the learning rate is 0.005.
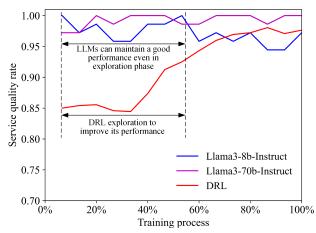
2) **Network Prediction**: The evaluation dataset is a publicly available network traffic dataset from the city of Milan [14], recording the time of user interactions and the base stations managing these interactions. The original grid of 100×100 base stations is reorganized into $25 \times 25 = 625$ aggregated base stations. Each aggregated base station covers an area of approximately $1km \times 1km$, including 8,923 samples. For the evaluation, we randomly select one base station, partitioning the dataset such that 70% is used for training, 10% for validation, and the remaining 20% for testing. The simulation algorithms include: ① **Self-refinement**: Our proposed self-refined prompting utilizes GPT-4 as the foundation model for one-day-ahead traffic prediction using historical traffic from the previous day. It follows the proposed self-refined prompting techniques as defined in Fig. 3 and Section V. ② **LLM-related baselines**: We include GPT-4 and GPT-3.5 without self-refinement as two LLM-related baselines, better demonstrating the capabilities of self-refinement. ③ **Conventional baselines**: Autoregressive integrated moving average (ARIMA) and LSTM as two conventional baselines. Through extensive experiments, the optimal parameters for the ARIMA baseline model are determined including autoregressive ($p = 2$), differences ($d = 1$), and moving average ($q = 2$) components. We deploy LSTM as an optimal baseline, since it has been specifically trained on the target network traffic dataset. By contrast, **Self-refinement** and **LLM-related baselines** have no previous knowledge on the target task.
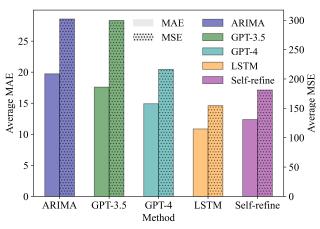
B. Simulation Results and Analyses

1) **Network Optimization Case Study**: Fig. 4(a) and 4(b) present the optimization task results. Specifically, one can observe that the average power consumption is constantly improved with the increasing number of episodes in Fig. 4(a). It indicates that the proposed scheme in Section IV enables LLMs to learn from previous examples and experience, and then improve its performance on target tasks iteratively. Fig. 4(a) also shows that LLMs present comparable performance as the DRL algorithm, which is achieved without model training or fine-tuning as in conventional ML algorithms. Meanwhile, Fig. 4(b) compares the service quality of Llama3-7b, Llama3-70b, and DRL, which means the probability of violating the preset average data rate threshold of 1.5 Mbps/per user. It highlights that LLMs can maintain high service quality at the beginning of the training process, while the DRL algorithm has a lower service quality in the initial exploration phase. Fig. 4(b) reveals that LLMs have a higher learning efficiency than the DRL algorithm. It can be explained by LLM's few-shot learning capabilities, in which LLMs can quickly learn from demonstrations and task descriptions to improve decisions. By contrast, the DRL approach relies on exploration and value
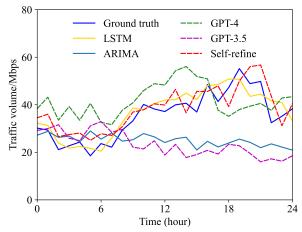
(a) **Optimization Case Study**: improved power consumption of Llamas.

(b) **Optimization Case Study**: service quality comparison with training.

(c) **Prediction Case Study**: average MAE and MSE comparison. The self-refine method achieves comparable performance as LSTM as a few-shot predictor, while LSTM has been delicately trained on the dataset.

(d) **Prediction Case Study**: predicted traffic comparison in 24 hours.

Fig. 4. Performance comparison in network optimization and traffic prediction case studies.

updating to improve the policy gradually, and the low sampling efficiency prevents the learning efficiency.

2) Network Prediction Case Study: Fig. 4(c) compares the average MAE and MSE metrics of various techniques. It is observed that the self-refined prompting method significantly outperforms ARIMA, GPT-3.5, and GPT-4, demonstrating the capabilities of our proposed self-refined prediction method. The self-refined prompting method exhibits notable improvements over GPT-3.5 and GPT-4, with MAE reductions of 29.72% and 17.09%, which underscores more powerful generalization capability on unseen datasets through iterative feedback generation and prediction refinement processes. GPT-3.5 shows a worse performance due to its outdated architecture and designs, which have been observed in many existing studies. Although LSTM serves as an optimal baseline with superior performance, it is important to note that LSTM is specifically trained on the target dataset, incurring additional

computational costs, which presents significant challenges in scenarios involving resource-constrained devices within wireless networks. Given the dynamic evolution of traffic distribution in non-stationary wireless networks, a well-trained LSTM may overfit the historical traffic used during the pre-training, resulting in degraded performance on new, unseen traffic over time. Furthermore, similar results can be observed in Fig. 4(d), which illustrates the one-day ground truth traffic and the predicted traffic of various methods. The self-refined prediction method more closely matches the ground truth traffic, indicating strong scalability of predictions across various base stations. By contrast, GPT-4 shows an obvious mismatch from 9:00 to 15:00.

In summary, the simulation results in Fig. 4 demonstrate that LLMs have great potential in handling network management, i.e., optimization and prediction tasks. They present satisfied performance as conventional machine learning algorithms such

as DRL and LSTM, while avoiding extra model training and fine-tuning complexity. LLMs also show higher learning efficiency than existing techniques, and they allow human language-based input, showing the possibility of natural language-based future network management.

VII. REAL-WORLD CHALLENGES AND LIMITATIONS

This section will further discuss the limitations and real-world challenges of prompt engineering applications.

Firstly, prompt engineering is inherently limited by the context window size of LLMs, which restricts the capabilities to process tasks requiring extensive contextual information beyond the model's input and memory constraints. Recent advancements such as Llama3 with a context window size of 128k tokens, significantly mitigate this limitation.

Secondly, the effectiveness of prompt engineering relies heavily on the selection of demonstrations shown to the model or the order of demonstrations. Poorly chosen or suboptimal examples can result in biased outputs. Our work considers iterative prompting to select better demonstrations, but professional experiences are still needed in this process.

Moreover, crafting effective prompts for complicated tasks in network environments usually requires trial-and-error experimentation. This iterative process is inefficient, especially when results are inconsistent or difficult to reproduce. Without clear guidelines, engineers must rely on intuition or extensive testing to achieve desired outcomes. This can slow down the development for novel and complex use cases as in 6G.

In addition, in real-world applications, prompt engineering may introduce significant security concerns, as it is potentially vulnerable to attacks such as prompt injection and leakage. Such attacks are particularly concerning in sensitive domains like wireless communications, where malicious actors could manipulate the model or extract encoded sensitive data [15].

Finally, note that prompt engineering is not a standalone solution but one part of pipelines of LLM and generative AI techniques [5]. Prompt engineering is constrained by the inherent limitations of the used LLMs. If the LLM lacks sufficient knowledge or context, prompting is unlikely to produce accurate results. Therefore, when prompt engineering struggles with highly specific tasks, it may require external support such as fine-tuning. This dependency highlights that prompt engineering can be actively combined with other techniques such as pre-training and fine-tuning to make the most of LLM potential.

VIII. CONCLUSION

The progress of generative AI and LLMs brings promising opportunities for next-generation wireless networks. This work provides a comprehensive overview of prompt engineering for network applications. It presents in-depth analyses of different prompting techniques such as in-context learning, chain-of-thought, and self-refinement. In addition, this work proposes two novel prompting schemes for network optimization and prediction problems, and the case studies show that the proposed methods can achieve satisfactory performance. In the future, we will explore more complicated and advanced network tasks such as reconfigurable intelligence surfaces and integrated sensing and communication, uncovering the full potential of LLMs for future wireless networks.

REFERENCES

[1] H. Zhou, C. Hu, Y. Yuan, Y. Cui, Y. Jin, C. Chen, H. Wu, D. Yuan, L. Jiang, D. Wu *et al.*, "Large language model (llm) for telecommunications: A comprehensive survey on principles, key techniques, and opportunities," *arXiv preprint arXiv:2405.10825*, 2024.

[2] C. Liang, H. Du, Y. Sun, D. Niyato, J. Kang, D. Zhao, and M. A. Imran, "Generative ai-driven semantic communication networks: Architecture, technologies and applications," *arXiv preprint arXiv:2401.00124*, 2023.

[3] S. Javaid, R. A. Khalil, N. Saeed, B. He, and M.-S. Alouini, "Leveraging large language models for integrated satellite-aerial-terrestrial networks: Recent advances and future directions," *arXiv preprint arXiv:2407.04581*, 2024.

[4] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," *arXiv:2402.07927*, 2024.

[5] H. Du, G. Liu, D. Niyato, J. Zhang, J. Kang, Z. Xiong, B. Ai, and D. I. Kim, "Generative AI-aided Joint Training-free Secure Semantic Communications via Multi-modal Prompts," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 12 896–12 900.

[6] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv:2303.18223*, 2023.

[7] Z. Lin, G. Qu, Q. Chen, X. Chen, Z. Chen, and K. Huang, "Pushing large language models to the 6g edge: Vision, challenges, and opportunities," *arXiv preprint arXiv:2309.16739*, 2023.

[8] M. Xu, D. Niyato, J. Kang, Z. Xiong, S. Mao, Z. Han, D. I. Kim, and K. B. Letaief, "When large language model agents meet 6g networks: Perception, grounding, and alignment," *IEEE Wireless Communications*, 2024.

[9] Y. Liu, H. Du, D. Niyato, J. Kang, S. Cui, X. Shen, and P. Zhang, "Optimizing mobile-edge ai-generated everything (aigx) services by prompt engineering: Fundamental, framework, and case study," *IEEE Network*, 2023.

[10] H. Zhou, C. Hu, D. Yuan, Y. Yuan, D. Wu, X. Liu, and C. Zhang, "Large language model (llm)-enabled in-context learning for wireless network optimization: A case study of power control," *arXiv preprint arXiv:2408.00214*, 2024.

[11] C. Hu, H. Zhou, D. Wu, X. Chen, J. Yan, and X. Liu, "Self-refined generative foundation models for wireless traffic prediction," *arXiv preprint arXiv:2408.10390*, 2024.

[12] P. Esling and C. Agon, "Time-series data mining," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1–34, 2012.

[13] M. Chiang, P. Hande, T. Lan, C. W. Tan *et al.*, "Power control in wireless cellular networks," *Foundations and Trends® in Networking*, vol. 2, no. 4, pp. 381–533, 2008.

[14] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, "A multi-source dataset of urban life in the city of Milan and the Province of Trentino," *Scientific data*, vol. 2, no. 1, pp. 1–15, 2015.

[15] H. Du, D. Niyato, J. Kang, Z. Xiong, K.-Y. Lam, Y. Fang, and Y. Li, "Spear or shield: Leveraging generative AI to tackle security threats of intelligent network services," *arXiv:2306.02384*, 2023.